OpenFox™

DMPP 2020 Workstation and Remote Agency Interface With Encryption

Version 6a

Texas Department of Public Safety TLETS

November 2, 2005
Revised June 1, 2007

The OpenFox Company

Computer Projects of Illinois, Inc.
6416 South Cass Avenue
Westmont, IL 60559
(630) 968-0244

TABLE OF CONTENTS

1. DOCUMENT PURPOSE

The purpose of the Interface Specifications DMPP 2020 / NCIC 2000 is to update both the communications facilities between the new TLETS message switching system that will be implemented by the Texas Department of Public Safety (TxDPS) and the remote interface agencies and to offer access to the enhanced operational systems of TLETS, NCIC and NLETS. The new TLETS interface standard will enable the TLETS system and its subscriber agencies to adopt a network protocol and interface specification that adheres to the national standards for the exchange of law enforcement and criminal justice information.

Inclusion of a format in this document does not mean that the format or the transaction will be supported indefinitely. TxDPS retains the right to make changes to this document as deemed necessary.

1.1. Background

The objective of the TLETS Re-engineering project is the replacement of the legacy TLETS message switch and legacy workstations, and the migration of TLETS subscribers to TCP/IP. The legacy message switch will be replaced by an OpenFox™ message switch provided by Computer Projects of Illinois (CPI). The legacy workstations will be replaced by Omni*xx*™ Force Browser client software provided by the Datamaxx Group. DMPP 2020 is the native protocol used by the Omni*xx*™ *Force* clients and will be the standard for interface agency connections.

In order to provide a migration path for legacy interface agency connections, CPI and Datamaxx are providing local agencies with the specifications and technical documentation for DMPP 2020.

1.2. Acknowledgements

OpenFox™ is a registered trademark of Computer Projects of Illinois.

"Datamaxx Message Processing Protocol" and "DMPP 2020" are registered trademarks of Datamaxx Applied Technologies, Inc. The information contained in this document was compiled based upon their description of the protocol and their implementations in several states that support both Omni*xx*™ Workstations and remote systems using DMPP 2020.

Detailed information can be obtained from documentation supplied by Datamaxx Applied Technologies, Inc.

2. GENERAL DESCRIPTION

This architectural specification provides software interface developers with a common reference for interoperable text and image manipulation on the new TLETS OpenFox™ system. The formats documented in the NCIC Message Book, Volume I, the NCIC 2000 Operating Manual (including subsequent Technical Operational Updates or TOUs) and the NLETS User and Technical Guide as published by the FBI, NCIC and NLETS are supported. Formats for Texas specific transactions, as published by TxDPS are also supported. The original document with changes will be in the possession of TxDPS and the latest updates will be noted.

The specification for this interface will be modeled after, but will not duplicate the current specifications for existing Interface Agencies. Characteristics of the interface are as follows:

1) TCP/IP over Ethernet
2) Full-duplex and asynchronous in operation.
3) Single socket operation utilizing port
4) Binary data is allowed.
5) The DMPP 2020 protocol utilizes a “keep-alive” message transmission which must be sent by the Remote Agency every 30 seconds if there is no intervening live traffic. A live traffic transmission serves the dual purpose of sending data as well as performing the “keep alive” function. If the Remote Agency does not send a “keep alive” for two consecutive 30-second intervals, the device will be marked as offline and in a “failed” condition.
6) The DMPP 2020 protocol uses message acknowledgements as an application level acknowledgement. The acknowledgement signifies that the message has been received, safely stored, and may be de-queued. The OpenFox™ message switch and the Remote Agency are required to acknowledge message transmissions immediately. Absence of a message acknowledgement requires that the message be resent. <u>NOTE: Receipt of a message acknowledgement before sending the next message in queue is not a requirement. This applies to the remote node as well as to OpenFox™.</u>
7) Each message transmission shall be referred to as a message block. Each message block shall contain multiple elements. The elements of a message block are: START PATTERN (STAP), BLOCK LENGTH, HEADER, DATA, STOP PATTERN (STOP).
    a. The START PATTERN is a sequence of 4 bytes (x’FF00AA55’) and indicates the start of a message transmission. This is the logical equivalent of the STX for the message transmission.
    b. The BLOCK LENGTH immediately follows the START PATTERN. BLOCK LENGTH is an unsigned, double-word, 32-bit, binary integer. The BLOCK LENGTH shall be the entire length of the message including START PATTERN, BLOCK LENGTH and STOP PATTERN.

c. The HEADER immediately follows the BLOCK LENGTH. The HEADER is 16 bytes in length and composed of multiple fields. The HEADER contains, HEADER LENGTH, FUNCTION, VALIDATION FIELD, DATA LENGTH, STATUS CODE, DESTINATION.

   i. HEADER LENGTH is a single word signed integer. The length of the HEADER is a constant 16 bytes and represented as (x'0010')

   ii. FUNCTION is a single word signed integer and can contain one of eight (8) entries. The possible values for FUNCTION are described in Section 3.5.

   iii. VALIDATION FIELD is a 4 byte sequence comprised of any characters. An acknowledgement will contain the same four characters.

   iv. DATA LENGTH is a double-word, 32-bit, signed integer which represents the length of the DATA element. As a quality control check, the DATA LENGTH element must be exactly 28 bytes less than the BLOCK LENGTH.

   v. STATUS CODE is a single-word, 16-bit signed integer

   vi. DESTINATION is single-word, 16-bit signed integer. All outgoing messages from OpenFox™ to the Remote Agency will contain a (x'0001').

d. The DATA element will contain the payload of the transaction, which is wrapped in the DMPP 2020 envelope. Message transmissions originated by the Remote Agency (inbound to the OpenFox™) will contain the following three mandatory fields prior to the message key: DAC, REFERENCE, USER-ID. The message key of the transaction will immediately follow the terminator of the USER-ID field.

   i. The DAC (Device Access Code) field is variable in length, with a maximum of 11 ASCII alphanumeric characters and represents the ORI or name of the device from which the transaction is originating. The DAC is not space filled and must be terminated by a 'period'.

   ii. The REFERENCE field is variable in length having a maximum of 10 ASCII alphanumeric characters, is the 'control field' or 'interface header' for this protocol. The REFERENCE field is not space filled and must be terminated by a 'period'.

   iii. The USER-ID is a filed length field that contains the 7 ASCII Character User ID assigned by TxDPS. The USER-ID is not space filled and must be terminated by a period.

e. The STOP PATTERN consisting of 4 bytes (x'55AA00FF') shall immediately follow the last character of the DATA element. The STOP PATTERN serves as the indicator of the end of a message transmission. This is the logical equivalent of the ETX for this protocol.

8) Responses from the OpenFox™ shall include all of the fields mentioned in #7, above.
9) Responses to inquiry or updates will always contain the original DAC, REFERENCE and User ID field that was input in the original transaction.
10) Unsolicited messages, such as administrative messages, may contain 'null' reference field if no control field was input by the originator. The User ID field will also be "null" if the message is an unsolicited message from out of state. Both fields will be terminated by a period.
11) Line terminators must always be an ASCII carriage-return (x'0D') followed by an ASCII line-feed (x'0A').

The following are general formats for message types to and from OpenFox™ via this interface.

2.1. Keep Alive Messages

Keep-alive messages shall be formatted according the DMPP 2020 specification as described in Section 4 of this document.

2.2. Message Acknowledgements

Message Acknowledgements shall be formatted according the DMPP 2020 specification Section 4 of this document. Receipt of an application level message acknowledgement is not required prior to sending the next message in queue.

2.2.1. Intermediate Block Acknowledgements

Requests for intermediate block acknowledgements are permissible, however OpenFox will not request such acknowledgements for multi-block messages. As an example, if a message spans 3 DMPP2020 blocks, the message would have the function codes set as follows:

Block1: MORE (function code 3)
Block2: MORE (function code 3)
Block3: MESGACK (function code 2)

2.3. Messages to TLETS OpenFox™ From Remote Agencies

The general format, not including the DMPP 2020 envelope, for messages input to TLETS OpenFox™ is as follows:

DAC.REFERENCE.USER-ID.MKE.<variable data>

Field	Length	Description
DAC	1-11	Device Name (ORI or Mnemonic)

Field	Length	Description
.	1	ASCII Period character.
REFERENCE	1-10	Control Field
.	1	ASCII Period character.
USER-ID	7	Assigned User Identification
.	1	ASCII Period character.
MKE	2 to 6	Message Key
.	1	ASCII Period character.
<variable data>	variable	Message text, printable ASCII characters according to MKE format requirements.

2.4. Messages From TLETS OpenFox™ to Remote Agencies

The general format, not including the DMPP 2020 envelope, for messages, including error messages, from (sent by) TLETS OpenFox™ is as follows:

DAC.REFERENCE.USER-ID.MKE.<variable data>

Field	Length	Description
DAC	1-11	Device Name
.	1	ASCII Period character.
REFERENCE	1-10	Control Field
.	1	ASCII Period character.
USER-ID	7	Assigned User Identification
.	1	ASCII Period character.
MKE	2 to 6	Message Key
.	1	ASCII Period character.
<variable data>	variable	Message text, printable ASCII characters according to MKE format requirements.

3. DMPP 2020 MESSAGE BLOCK

3.1. General Description

The DMPP 2020 message block is comprised of the following five (5) areas:

STAP	=	Start Pattern
Block length	=	Overall message length of entire message block
Header	=	Comprised of six (6) elements as described in section 1.2
Message Data	=	Variable text comprising the data "payload" of the transaction
STOP	=	Stop Pattern

STAP (4 bytes)	**Block Length (4 bytes)**	**Header (variable)**	**[Message] Data (variable)**	**STOP (4 bytes)**
FF00AA55	<binary block length>	<see below>	<message data>	Hex = 55AA00FF

3.2. "Header" Area Breakdown (No Encryption)

Header Length (2 bytes)	**Function (2 bytes)**	**Validation Field (4 bytes)**	**Data Length (4 bytes)**	**Status Code (2 bytes)**	**Destination (2 bytes)**
16 bit signed integer VALUE = 16 (x0010)	16 bit signed integer VALUE (hex) = 0001, 0002, 0003, 0004, 0011, 0012, 0021, 0022	32 bit unsigned integer (content returned by OpenFox™)	32 bit signed integer (see Note 1)	16 bit signed integer	16 bit signed integer VALUE = 0001 on Outgoing Msgs & Ignored on inbound Msgs

Note 1: The BLOCK LENGTH must be greater than DATA LENGTH by the amount equal to the length of the DMPP 2020 envelope which is 28 bytes total in this case.

3.3. "Header" Area Breakdown (Encryption)

Header Length (2 bytes)	Function (2 bytes)	Validation Field (4 bytes)	Data Length (4 bytes)	Status Code (2 bytes)	Destination (2 bytes)
16 bit signed integer VALUE = 42 (x002A)	16 bit signed integer VALUE (hex) = 0001, 0002, 0003, 0004, 0011, 0012, 0021, 0022	32 bit unsigned integer (content returned by OpenFox™)	32 bit signed integer (see Note 2) (Encrypted Length)	16 bit signed integer VALUE (hex) = 01,02,03,04	16 bit signed integer VALUE = 0001 on Outgoing Msgs & Ignored on inbound Msgs

Encryption Length (2 bytes)	Function (2 bytes)	Book Id (2 bytes)	Key Id (2 bytes)	CRC-16 (2 bytes)	AES Initialization Vector (16 bytes)
16 bit signed integer VALUE = 26 (x001A)	16 bit signed integer VALUE (hex) = 0001, 0002	16 bit unsigned integer	16 bit unsigned integer	16 bit value	16 characters

Note 2: The BLOCK LENGTH must be greater than DATA LENGTH by the amount equal to the length of the DMPP 2020 envelope which is 54 bytes total in this case.

3.4. Message Data

The message data area will immediately following the "Header" fields and will precede the STOP pattern. For acknowledgements and "Keep Alive" messages the data length is zero so no data is present.

Message Data

3.5. Message Header Fields

There are six required and six optional (encryption) fields in the message header, which are all used by OpenFox™. The fields, and the appropriate values, appear below.

Header Length	This field is set to 16 (hex 0010) or 42 (hex 002A) depending on encryption setting.	
Function	The functions supported are: (hex)	
	0001	Data message with no acknowledgment, final block
	0002	Data message with acknowledgment, final block
	0003	Data message with no acknowledgment, more blocks to follow (see note below)
	0004	Data message with acknowledgment, more blocks to follow (see note below)
	0011	Positive acknowledgment to data message (Status Code is set to "Successful receipt of data message")
	0012	Negative acknowledgment to data message (Error is defined in the Status Code field)
	0021	Request status of system
	0022	Response to status request (Status is defined in the Status Code field)
	0031	Send Coded Message 1
	0032	Send Coded Message 2
	0041	Positive response to Coded Message 1
	0042	Positive response to Coded Message 2
Validation	The contents of this field are returned by OpenFox™.	
Data Length	This field represents the data length as an unsigned 32-bit number. No single block may be larger than 32K.	
Status	The OpenFox™ uses this field as documented in the DMPP-2020 specification.	
	The status codes for request messages are (hex):	
	0001	Message does not contain binary object
	0002	Message contains binary object in NCIC transaction format
	0003	Message contains binary object in NCIC response format
	0004	Message contains binary object in DSEO-20202 format
	The status codes for response messages are (hex):	
	0001	Successful receipt of data message
	0011	Permanent (i.e., non-recoverable) error occurred (e.g. disk failure)
	0012	Temporary (i.e., recoverable) error occurred (e.g. printer out of paper)

	0013	Logical error occurred (e.g. too many messages received too quickly, and thus a queue containing acknowledgments filled up)
	0014	Message length exceeds maximum, message will be discarded
	0021	Queried destination is available and ready
	0022	Queried destination is available, but not ready (e.g. printer has buffer space, but is out of paper)
	0023	Queried destination is not available and not ready
	0041	Invalid function code received
	0042	Invalid (or non-existent) destination received
	0043	Invalid Extended Message Header format or length received
	0044	Function not supported
Destination	The value is always set to hex 0001 on outgoing messages, and ignored on inbound messages	
Encryption Length	This field is set to 26 (hex 001A) when present.	
Function	The functions supported are:	
	0001	AES Encryption with 128 bit keys, 128 bit blocks, CBC mode
	0002	AES Encryption with 256 bit keys, 128 bit blocks, CBC mode
Book Id	Contains the Key Book number.	
Data Length	Contains the Key Id within the book.	
CRC-16	Contains the 2-character CRC-16 or the original unencrypted data.	
Init Vector	Contains the 16-character AES random initialization vector.	

3.6. Message Guidelines

1 The end point workstation or remote server must initiate the connection, using a "connect" with the IP address and Port number provided by OpenFox™.

2 Either end may break the connection at any time. The end point should wait a short period (e.g. 30 seconds) when attempting to re-connect. If initial connect fails, the end point should wait a short period (e.g. 30 seconds) before re-issuing the connect.

3 The end point should wait for the DMPP "ACK" from a message before sending the next one.

4 The end point should not consider a sent message delivered until the DMPP "ACK" is received from OpenFox™.

5 The end point should store a received message safely before sending the DMPP "ACK" to OpenFox™. Store time should not exceed 1 second from receipt of message to the send of the "ACK".

6 Observe "Network Byte Order" rules for the integer fields in the DMPP Message Header.
7 Initialize all header fields correctly.
8 The "keep alive" time is based on last activity -- it is NOT a periodic clock time.
9 Scanning for the end pattern is not enough to ensure block integrity, as that pattern can appear legally in images. The length field MUST also be used.
10 The end pattern can be broken across received data blocks. Data must be read as a character stream.
11 The block length field for the entire DMPP message block includes itself and the start and stop patterns
12 General received packet process:
  12.a Search for "Start" Pattern
  12.b Calculate Length from Next 4 Characters
  12.c Block Length includes the "Start" and "End" Patterns as well as the length of the Block Length field (12 Character Overhead)
  12.d Accumulate Until All Length Exhausted
  12.e Verify "End Pattern"
  12.f Process header and then data (if any)

4. "KEEP ALIVE" MESSAGES

The OpenFox™ uses the status request/response function to act as an idle line timer. OpenFox™ will terminate a connection that has had no activity for 60 seconds which allows for a single "missed" keep alive message.

4.1. DMPP 2020 "Keep Alive" From Workstation or Server

(30 seconds from last activity)

STAP **(4 bytes)**	**block length** **(4 bytes)**	**Header** **(16 bytes)**	**[Message] Data** **(variable**	**STOP** **(4 bytes)**
Hex = FF00AA55	Hex = 0000 001C	<see below>	<message data>	Hex = 55AA00FF

Header Length **(2 bytes)**	**Function** **(2 bytes)**	**Validation Field** **(4 bytes)**	**Data Length** **(4 bytes)**	**Status Code** **(2 bytes)**	**Destination** **(2 bytes)**
Hex = 0010	Hex = 0021	Hex = 3136 3138	Hex = 0000 0000	HEX = 0021	Hex = 0001

Message Data
(no data)

4.2. Response To Keep-Alive By OpenFox™

STAP (4 bytes)	**block length (4 bytes)**	**Header (16 bytes)**	**[Message] Data (variable**	**STOP (4 bytes)**
Hex = FF00AA55	Hex 0000 001C	<see below>	<message data>	Hex = 55AA00FF

Header Length (2 bytes)	**Function (2 bytes)**	**Validation Field (4 bytes)**	**Data Length (4 bytes)**	**Status Code (2 bytes)**	**Destination (2 bytes)**
Hex = 0010	Hex = 0022	Hex = 3136 3138	Hex = 0000 0000	Hex = 0021	Hex = 0001

Message Data
(no data)

5. OMN*IXX*™ WORKSTATION MESSAGES

5.1. NLETS Message Example w/ Encryption

STAP **(4 bytes)**	**block length** **(4 bytes)**	**Header** **(42 bytes)**	**[Message] Data** **(variable**	**STOP** **(4 bytes)**
Hex = FF00AA55	Hex = 0000 0237	<see below>	<message data>	Hex = 55AA00FF

Header Length **(2 bytes)**	**Function** **(2 bytes)**	**Validation Field** **(4 bytes)**	**Data Length** **(4 bytes)**	**Status Code** **(2 bytes)**	**Destination** **(2 bytes)**
Hex = 002A	Hex = 0002	Hex = 3231 3230	Hex = 0000 020D	Hex = 0001	VALUE = 0001

Encryption Length **(2 bytes)**	**Function** **(2 bytes)**	**Book Id** **(2 bytes)**	**Key Id (2 bytes)**	**CRC-16** **(2 bytes)**	**AES Initialization** **Vector (16 bytes)**
16 bit signed integer VALUE = 26 (x001A)	16 bit signed integer VALUE (hex) = 0002	16 bit unsigned integer	16 bit unsigned integer	16 bit value	16 characters

Message Data

`)((&^**&%^**ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^**&(yh)()((&^**&%^**ih(&*^(**y(j(*(y(*&h(&*^(**y`
`(j(*(y(*&(&*(uj()*^&*^**&(yh)(((&^**&%^**ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^**&(yh)()((&^**&%^**`
`ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^**&(yh)(((&^**&%^**ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&`
`(uj()*^&*^**&(yh)()((&^**&%^**ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^**&(yh)(((&^**&%^**ih(&*^(**y(j`
`(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^**&(yh)()((&^**&%^**ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^`

5.2. NCIC Message Example w/ Encryption

STAP (4 bytes)	**block length (4 bytes)**	**Header (42 bytes)**	**[Message] Data (variable**	**STOP (4 bytes)**
Hex = FF00AA55	Hex = 0000 0237	<see below>	<message data>	Hex = 55AA00FF

Header Length (2 bytes)	**Function (2 bytes)**	**Validation Field (4 bytes)**	**Data Length (4 bytes)**	**Status Code (2 bytes)**	**Destination (2 bytes)**
Hex = 002A	Hex = 0002	Hex = 3231 3230	Hex = 0000 020D	Hex = 0001	VALUE = 0001

Encryption Length (2 bytes)	**Function (2 bytes)**	**Book Id (2 bytes)**	**Key Id (2 bytes)**	**CRC-16 (2 bytes)**	**AES Initialization Vector (16 bytes)**
16 bit signed integer VALUE = 26 (x001A)	16 bit signed integer VALUE (hex) = 0002	16 bit unsigned integer	16 bit unsigned integer	16 bit value	16 characters

Message Data
`)((&^*&%^*ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^*&(yh)()((&^*&%^*ih(&*^(**y(j(*(y(*&h(&*^(**y`
`(j(*(y(*&(&*(uj()*^&*^*&(yh)(((&^*&%^*ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^*&(yh)()((&^*&%^*`
`ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^*&(yh)(((&^*&%^*ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&`
`(uj()*^&*^*&(yh)()((&^*&%^*ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^*&(yh)(((&^*&%^*ih(&*^(**y(j`
`(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^*&(yh)()((&^*&%^*ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^`

5.3. Acknowledgement From OpenFox™

Note: Acknowledgements are not encrypted because there is no data.

STAP (4 bytes)	**block length (4 bytes)**	**Header (16 bytes)**	**[Message] Data (variable**	**STOP (4 bytes)**
hex = FF00AA55	Hex = 0000 001C	<see below>	<message data>	Hex = 55AA00FF

Header Length (2 bytes)	**Function (2 bytes)**	**Validation Field (4 bytes)**	**Data Length (4 bytes)**	**Status Code (2 bytes)**	**Destination (2 bytes)**
Hex = 0010	Hex = 0011	Hex = 3231 3230	Hex = 0000 0000	Hex = 0001	Hex = 0001

Message Data
(no data)

5.4. NLETS Response To Workstation w/ Encryption

STAP (4 bytes)	block length (4 bytes)	Header (42 bytes)	[Message] Data (variable	STOP (4 bytes)
HEX = FF00AA55	Hex = 0000 0237	<see below>	<message data>	55AA00FF

Header Length (2 bytes)	Function (2 bytes)	Validation Field (4 bytes)	Data Length (4 bytes)	Status Code (2 bytes)	Destination (2 bytes)
Hex = 002A	Hex = 0002	Hex = 3231 3230	Hex = 0000 020D	Hex = 0001	VALUE = 0001

Encryption Length (2 bytes)	Function (2 bytes)	Book Id (2 bytes)	Key Id (2 bytes)	CRC-16 (2 bytes)	AES Initialization Vector (16 bytes)
16 bit signed integer VALUE = 26 (x001A)	16 bit signed integer VALUE (hex) = 0002	16 bit unsigned integer	16 bit unsigned integer	16 bit value	16 characters

Message Data
)((&^*&%^**ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^*&(yh)()((&^*&%^**ih(&*^(**y(j(*(y(*&h(&*^(**y (j(*(y(*&(&*(uj()*^&*^*&(yh)((((&^*&%^**ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^*&(yh)()((&^*&%^** ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^*&(yh)((((&^*&%^**ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(& (uj()*^&*^*&(yh)()((&^*&%^**ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^*&(yh)((((&^*&%^**ih(&*^(**y(j (*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^*&(yh)()((&^*&%^**ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^

5.5. NCIC Response To Workstation w/ Encryption

STAP (4 bytes)	**block length (4 bytes)**	**Header (42 bytes)**	**[Message] Data (variable**	**STOP (4 bytes)**
Hex = FF00AA55	Hex = 0000 0135	<see below>	<message data>	Hex = 55AA00FF

Header Length (2 bytes)	**Function (2 bytes)**	**Validation Field (4 bytes)**	**Data Length (4 bytes)**	**Status Code (2 bytes)**	**Destination (2 bytes)**
Hex = 002A	Hex = 0002	Hex = 3231 3230	Hex = 0000 010B	Hex = 0001	VALUE = 0001

Encryption Length (2 bytes)	**Function (2 bytes)**	**Book Id (2 bytes)**	**Key Id (2 bytes)**	**CRC-16 (2 bytes)**	**AES Initialization Vector (16 bytes)**
16 bit signed integer VALUE = 26 (x001A)	16 bit signed integer VALUE (hex) = 0002	16 bit unsigned integer	16 bit unsigned integer	16 bit value	16 characters

Message Data
`)((&^*&%^**ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^*&(yh)()((&^*&%^**ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^*&(yh)(((&^*&%^*ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^*&(yh)()((&^*&%^**ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^*&(yh)(`

6. REMOTE SERVER MESSAGES (NON-OFML)

The message data portion of the input message inbound to OpenFox™ from remote servers will be preceded by three "control" type fields. The fields are "DAC" (Device Address Code), "REFERENCE", and "USER-ID" The "DAC" field identifies the device from which the message came. That is the device behind the remote interface. "REFERENCE" is the field similar to the NLETS or NCIC control field and will be returned to the remote server in any response. This field many times is used for remote server routing purposes. The "USER-ID" field identifies the user behind the remote server that is operating the remote workstation or device. All fields are required.

• DAC =	Max. 11 Alpha-Num
• REFERENCE=	Max. 10 Alpha-Num + special characters as defined by NLETS and NCIC
• USER-ID=	Min 7 / Max. 7 Alpha-Num

6.1. NLETS Message Example w/o Encryption

STAP (4 bytes)	**block length (4 bytes)**	**Header (16 bytes)**	**[Message] Data (variable**	**STOP (4 bytes)**
HEX = FF00AA55	Hex = 0000 0060	<see below>	<message data>	55AA00FF

Header Length (2 bytes)	**Function (2 bytes)**	**Validation Field (4 bytes)**	**Data Length (4 bytes)**	**Status Code (2 bytes)**	**Destination (2 bytes)**
Hex = 0010	Hex = 0002	Hex = 3231 3230	Hex = 0000 0050	Hex = 0001	VALUE = 0001

Message Data
`XX048063Y.KAVA85880.RM6871.RQ.TX0100000.IL.TXT{0D0A}NAM/RECORD,TEST.DOB/19430504`

6.2. NLETS Message Example w/Encryption

STAP (4 bytes)	**block length (4 bytes)**	**Header (42 bytes)**	**[Message] Data (variable**	**STOP (4 bytes)**
Hex = FF00AA55	Hex = 0000 00ED	<see below>	<message data>	Hex = 55AA00FF

Header Length (2 bytes)	**Function (2 bytes)**	**Validation Field (4 bytes)**	**Data Length (4 bytes)**	**Status Code (2 bytes)**	**Destination (2 bytes)**
Hex = 002A	Hex = 0002	Hex = 3231 3230	Hex = 0000 00C3	Hex = 0001	VALUE = 0001

Encryption Length (2 bytes)	**Function (2 bytes)**	**Book Id (2 bytes)**	**Key Id (2 bytes)**	**CRC-16 (2 bytes)**	**AES Initialization Vector (16 bytes)**
16 bit signed integer VALUE = 26 (x001A)	16 bit signed integer VALUE (hex) = 0002	16 bit unsigned integer	16 bit unsigned integer	16 bit value	16 characters

Message Data[1]
`h(&*^(**y(j(*(y(*&(&*(uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^*&(yh)( (&^*&%^*ih(&*^(**y(j(*(y(*&h(&*^(**y(j(*(y(*&(&*(uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&`

[1] {0D0A} is representative of the ASCII carriage return, line feed pair. Brackets are not included

6.3. NCIC Message Example w/o Encryption

STAP (4 bytes)	block length (4 bytes)	Header (16 bytes)	[Message] Data (variable	STOP (4 bytes)
Hex = FF00AA55	Hex = 0000 0055	<see below>	<message data>	Hex = 55AA00FF

Header Length (2 bytes)	Function (2 bytes)	Validation Field (4 bytes)	Data Length (4 bytes)	Status Code (2 bytes)	Destination (2 bytes)
Hex = 0010	Hex = 0002	Hex = 3231 3230	Hex = 0000 0045	Hex = 0001	VALUE = 0001

Message Data
XX048063Y.KAVA85880.1234567.QW.TX0100000.NAM/RECORD,TEST.DOB/19430504

6.4. NCIC Message Example w/Encryption

STAP (4 bytes)	block length (4 bytes)	Header (42 bytes)	[Message] Data (variable	STOP (4 bytes)
Hex = FF00AA55	Hex = 0000 00EA	<see below>	<message data>	Hex = 55AA00FF

Header Length (2 bytes)	Function (2 bytes)	Validation Field (4 bytes)	Data Length (4 bytes)	Status Code (2 bytes)	Destination (2 bytes)
Hex = 002A	Hex = 0002	Hex = 3231 3230	Hex = 0000 00C0	Hex = 0001	VALUE = 0001

Encryption Length (2 bytes)	Function (2 bytes)	Book Id (2 bytes)	Key Id (2 bytes)	CRC-16 (2 bytes)	AES Initialization Vector (16 bytes)
16 bit signed integer VALUE = 26 (x001A)	16 bit signed integer Hex = 0002	16 bit unsigned integer	16 bit unsigned integer	16 bit value	16 characters

Message Data

```
(&*%ih)&*^(hoh)&*&^*hoij(*&^*^*(yho&*&^*ho&*(*&^(u&(&*(uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&
(&*(uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&(&*(uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&
```

7. REMOTE SERVER MESSAGES (OFML)

The message data portion of the input message inbound to OpenFox™ from remote servers can optionally use the OFML format, as used by the workstations. The Trusted server "control" type fields will be defined in the header, as follows:

• DAC =	Max. 11 Alpha-Num (OFML.HDR.DAC Element)
• REFERENCE=	Max. 10 Alpha-Num (OFML.HDR.REF Element)
• USER-ID=	Max. 30 Alpha-Num (OFML.HDR.USR Element)

7.1. NLETS Message Example (OFML) w/o Encryption

STAP (4 bytes)	**block length (4 bytes)**	**Header (16 bytes)**	**[Message] Data (variable**	**STOP (4 bytes)**
HEX = FF00AA55	Hex = 0000 011C	<see below>	<message data>	55AA00FF

Header Length (2 bytes)	**Function (2 bytes)**	**Validation Field (4 bytes)**	**Data Length (4 bytes)**	**Status Code (2 bytes)**	**Destination (2 bytes)**
Hex = 0010	Hex = 0002	Hex = 3231 3230	Hex = 0000 010C	Hex = 0001	VALUE = 0001

Message Data

```
<OFML><HDR><ID>014S000003</ID><DAC>SPHQ00P6</DAC><DAT>20040108073526</DAT><USR>1234567</USR>
<REF>014S000003</REF><MKE>RQ</MKE><ORI>TX0000000</ORI><SUM>RQLIC: KEATON, IL</SUM><DST
EID="DRI">IL</DST></HDR><TRN><LIC>TEST1234</LIC><LIY>2003</LIY>
<LIT>PC</LIT></TRN></OFML>
```

7.2. NLETS Message Example (OFML) w/Encryption

STAP (4 bytes)	**block length (4 bytes)**	**Header (42 bytes)**	**[Message] Data (variable**	**STOP (4 bytes)**
HEX = FF00AA55	Hex = 0000 0214	<see below>	<message data>	55AA00FF

Header Length (2 bytes)	**Function (2 bytes)**	**Validation Field (4 bytes)**	**Data Length (4 bytes)**	**Status Code (2 bytes)**	**Destination (2 bytes)**
Hex = 002A	Hex = 0002	Hex = 3231 3230	Hex = 0000 01EA	Hex = 0001	VALUE = 0001

Encryption Length (2 bytes)	**Function (2 bytes)**	**Book Id (2 bytes)**	**Key Id (2 bytes)**	**CRC-16 (2 bytes)**	**AES Initialization Vector (16 bytes)**
16 bit signed integer VALUE = 26 (x001A)	16 bit signed integer VALUE (hex) = 0002	16 bit unsigned integer	16 bit unsigned integer	16 bit value	16 characters

Message Data

```
(&*%ih)&*^(hoh)&*&^*hoij(*&^*^*(yho&*&^*ho&*(*&^(u&(&*(uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&(&*%ih
)&*^(hoh)&*&^*hoij(*&^*^*(yho&*&^*ho&*(*&^(u&(&*(uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&(&*%ih)&*^(h
oh)&*&^*hoij(*&^*^*(yho&*&^*ho&*(*&^(u&(&*(uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&(&*%ih)&*^(hoh)&*&
^*hoij(*&^*^*(yho&*&^*ho&*(*&^(u&(&*(uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&(&*%ih)&*^(hoh)&*&^*hoij
(*&^*^*(yho&*&^*ho&*(*&^(u&(&*(uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&
```

7.3. NCIC Message Example (OFML) w/o Encryption

STAP (4 bytes)	block length (4 bytes)	Header (16 bytes)	[Message] Data (variable	STOP (4 bytes)
HEX = FF00AA55	Hex = 0000 010A	<see below>	<message data>	55AA00FF

Header Length (2 bytes)	Function (2 bytes)	Validation Field (4 bytes)	Data Length (4 bytes)	Status Code (2 bytes)	Destination (2 bytes)
Hex = 0010	Hex = 0002	Hex = 3231 3230	Hex = 0000 00FA	Hex = 0001	VALUE = 0001

Message Data

`<OFML><HDR><ID>014S000008</ID><DAC>SPHQ00P6</DAC><DAT>20040108073650</DAT>`
`<REF>014S000008</REF><USR>1234567</USR><MKE>QV</MKE><ORI>TX0000000</ORI><SUM>QV: KEATON, </SUM>`
`</HDR><TRN><LIC>TEST1234</LIC><LIS>IN</LIS><RSH>Y</RSH><HIT>10</HIT></TRN></OFML>`

7.4. NCIC Message Example (OFML) w/Encryption

STAP (4 bytes)	block length (4 bytes)	Header (42 bytes)	[Message] Data (variable	STOP (4 bytes)
HEX = FF00AA55	Hex = 0000 0144	<see below>	<message data>	55AA00FF

Header Length (2 bytes)	Function (2 bytes)	Validation Field (4 bytes)	Data Length (4 bytes)	Status Code (2 bytes)	Destination (2 bytes)
Hex = 002A	Hex = 0002	Hex = 3231 3230	Hex = 0000 011A	Hex = 0001	VALUE = 0001

Encryption Length (2 bytes)	Function (2 bytes)	Book Id (2 bytes)	Key Id (2 bytes)	CRC-16 (2 bytes)	AES Initialization Vector (16 bytes)
16 bit signed integer VALUE = 26 (x001A)	16 bit signed integer VALUE (hex) = 0002	16 bit unsigned integer	16 bit unsigned integer	16 bit value	16 characters

Message Data
`oh)&*&^**hoij(*&^*^*(yho&*&^*ho&*(*&^(u&(&*(uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&(&*%ih)&*^(hoh)&*&^**hoij(*&^*^*(yho&*&^*ho&*(*&^(u&(&*(uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&(&*%ih)&*^(hoh)&*&^**hoij(*&^*^*(yho&*&^*ho&*(*&^(u&(&*(uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&`

7.5. NLETS Response To Remote Server (OFML) w/o Encryption

Messages output to remote servers by OpenFox™ using DMPP 2020 will have the ”DAC”, “REFERENCE” and “USER-ID” fields incorporated in the OFML “HDR” element. For unsolicited messages, the reference field will be blank. For unsolicited messages from out-of-state, the User-ID will be blank as well.

STAP (4 bytes)	block length (4 bytes)	Header (16 bytes)	[Message] Data (variable	STOP (4 bytes)
HEX = FF00AA55	VALUE (hex) = 0000 0229	<see below>	<message data>	55AA00FF

Header Length (2 bytes)	Function (2 bytes)	Validation Field (4 bytes)	Data Length (4 bytes)	Status Code (2 bytes)	Destination (2 bytes)
Hex = 0010	Hex = 0002	Hex = 3231 3230	Hex = 0000 0219	Hex = 0001	VALUE = 0001

Message Data

```
<OFML><HDR><ID>014S000003</ID><DAC>SPHQ00P6</DAC><SRC>NLET</SRC>
<DAT>20040108073344</DAT><REF>014S000003</REF><USR>1234567</USR><MKE>RR</MKE><ORI>ILLIC0000</ORI>
<DST>TX0000000</DST><DST>SPHQ00P6</DST><CTL>MRI3718976</CTL><SUM>RQLIC: KEATON, IL</SUM></HDR><RSP>
<TXT>RR.ILLIC0000{OD0A}05:35 01/08/2004 03911{OD0A}05:35 01/08/2004 01326
TX0000000{OD0A}*MRI3718976{OD0A}TXT{OD0A}010804  0632{OD0A}       LIC/TEST1234
INVALID{OD0A}LIC/TEST1234.LIT/PC{OD0A}MRI 3718979 IN: NLI1 1971 AT 08JAN2004 07:33:43{OD0A}OUT: SPHQ00P6 3
AT 08JAN2004 07:33:44{OD0A}</TXT></RSP></OFML>
```

Note 3: For responses to queries, the REFERENCE field will contain the data submitted in that field in the original inquiry

7.6. NCIC Response To Remote Server (OFML) w/Encryption

Messages output to Remote Servers by OpenFox™ using DMPP 2020 will have the "DAC", "REFERENCE" and "USER-ID" fields incorporated in the OFML "HDR" element.   For unsolicited messages (e.g. Delayed Hit Notification), the reference field will be blank. For unsolicited messages from out-of-state, the User-ID will be blank as well.

STAP **(4 bytes)**	**block length** **(4 bytes)**	**Header** **(42 bytes)**	**[Message] Data** **(variable**	**STOP** **(4 bytes)**
HEX = FF00AA55	VALUE (hex) = 0000 026A	<see below>	<message data>	55AA00FF

Header Length **(2 bytes)**	**Function** **(2 bytes)**	**Validation Field** **(4 bytes)**	**Data Length** **(4 bytes)**	**Status Code** **(2 bytes)**	**Destination** **(2 bytes)**
Hex = 002A	Hex = 0002	Hex = 3231 3230	Hex = 0000 0234	Hex = 0001	VALUE = 0001

Encryption Length **(2 bytes)**	**Function** **(2 bytes)**	**Book Id** **(2 bytes)**	**Key Id** **(2 bytes)**	**CRC-16** **(2 bytes)**	**AES Initialization** **Vector (16 bytes)**
16 bit signed integer VALUE = 26 (x001A)	16 bit signed integer VALUE (hex) = 0002	16 bit unsigned integer	16 bit unsigned integer	16 bit value	16 characters

Message Data
oh)&*&^*hoij(*&^*^*(yho&*&^*ho&*(*&^(u&(&*(uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&(&*%ih)&*^(hoh)&*& ^*hoij(*&^*^*(yho&*&^*ho&*(*&^(u&(&*(uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&(&*%ih)&*^(hoh)&*&^*hoij (*&^*^*(yho&*&^*ho&*(*&^(u&(&*(uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&oh)&*&^*hoij(*&^*^*(yho&*&^*ho &*(*&^(u&(&*(uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&(&*%ih)&*^(hoh)&*&^*hoij(*&^*^*(yho&*&^*ho&*(*&^ (u&(&*(uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&(&*%ih)&*^(hoh)&*&^*hoij(*&^*^*(yho&*&^*ho&*(*&^(u&(&* (uj()*^&*^*&(yh)((&^*&%^*ih(&*^(**y(j(*(y(*&

8. ENCRYPTION EXAMPLE

The following contains information to allow the remote agency to offline test and verify their encryption technique is working properly and will be compatible with the central message switch.

8.1. Example Key (256-bits)

The following 256-bit key is used is this example. The key is represented in HEX notation.

0x’331973EFE545A3CDEE0A87C9455F8584C02377692E0C2976345691CBE827CCF4’

8.2. Example IV (128-bits)

The following is the interrupt vector used in this example, again in HEX notation.

0x’2FC4DDF7D776241C733AA07FFA2BE456’

8.3. Test Message - OFML

The following example is an OFML formatted message (‘RQ’) for use in this test. The message is 210 bytes with no whitespace, no carriage returns, no linefeeds.

**<OFML><HDR><ID>89179</ID><DAC>IPS9</DAC><DAT>20050929141334</DAT><REF>89179</REF>
<MKE>RQ</MKE><USR>RON</USR><ORI>TX12345C1</ORI><DST>TX</DST></HDR><TRN><LIC>
ABC123</LIC><LIY>2005</LIY><LIT>PC</LIT></TRN></OFML>**

8.4. Calculated CRC

Submitting this text through the CRC algorithm yields a value of (HEX): 0x’7ED6’

8.5. Resulting Encrypted Message

Running this plain text through the encryption (apply PKCS#7 padding, run CBC mode, use above key and IV) yields the following 224 bytes of cipher text (in HEX):

**0x’14AE377E4B9737C9313CFE551E2401A6AAA28D88A3C29F375AFAFD7A820145B63F9E9A135C068F35
1A404C65FCE74594A991FF5E62883B599EC4B7C58A20822A7E960DBCF68E6585243A4D1EC68A93BA
A81514CBCEBB96BCF9C88E3C3C41B0F438C85ED4203DF0EEEF7087696A68937D285EF3BADC0B1910
AB0125D3A43F12EF7D246090F3435D2E839C6BA32F52DAA705E1AE8E65C3293CBC9D45012EE63989
B3A4A64ACD66EC4E1550FA39B35489762DBFA2431D56077622C2D2E0683FB94BD62F196FF6E74E39
1DA606662266D68548942628D1080BE1004188849AACE07A’**

A simple test to work backwards is to decrypt the above cipher text with the same key and IV, then remove the PKCS#7 padding. The result should be the original 210 bytes of plain text.